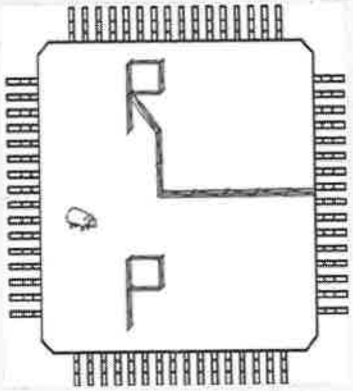
EXHIBIT B

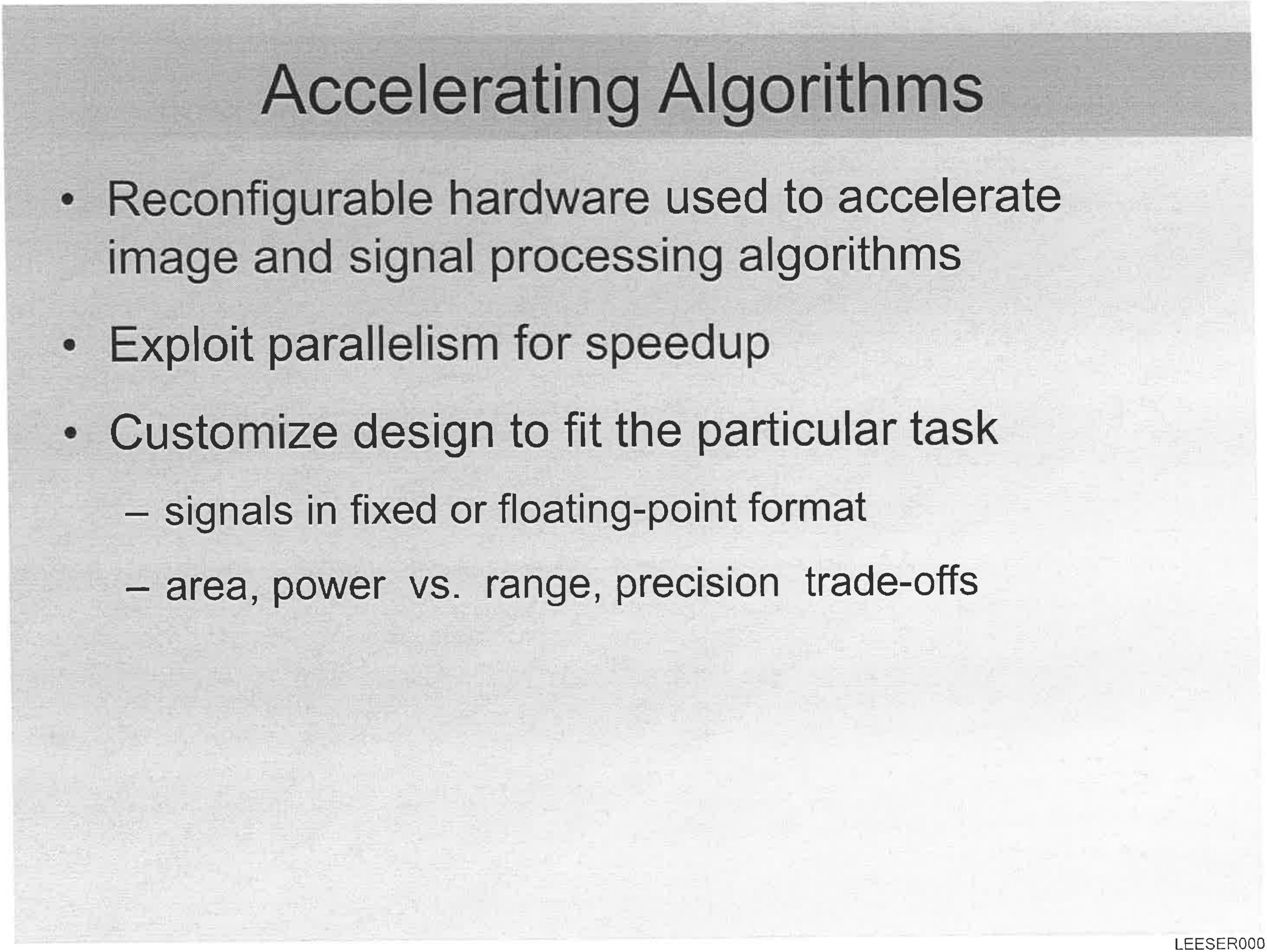
A Library of Parameterized Hardware Modules for Floating Point Arithmetic and Its Use

Prof. Miriam Leeser
Pavle Belanovic
Department of Electrical and Computer Engineering
Northeastern University
Boston MA



Outline

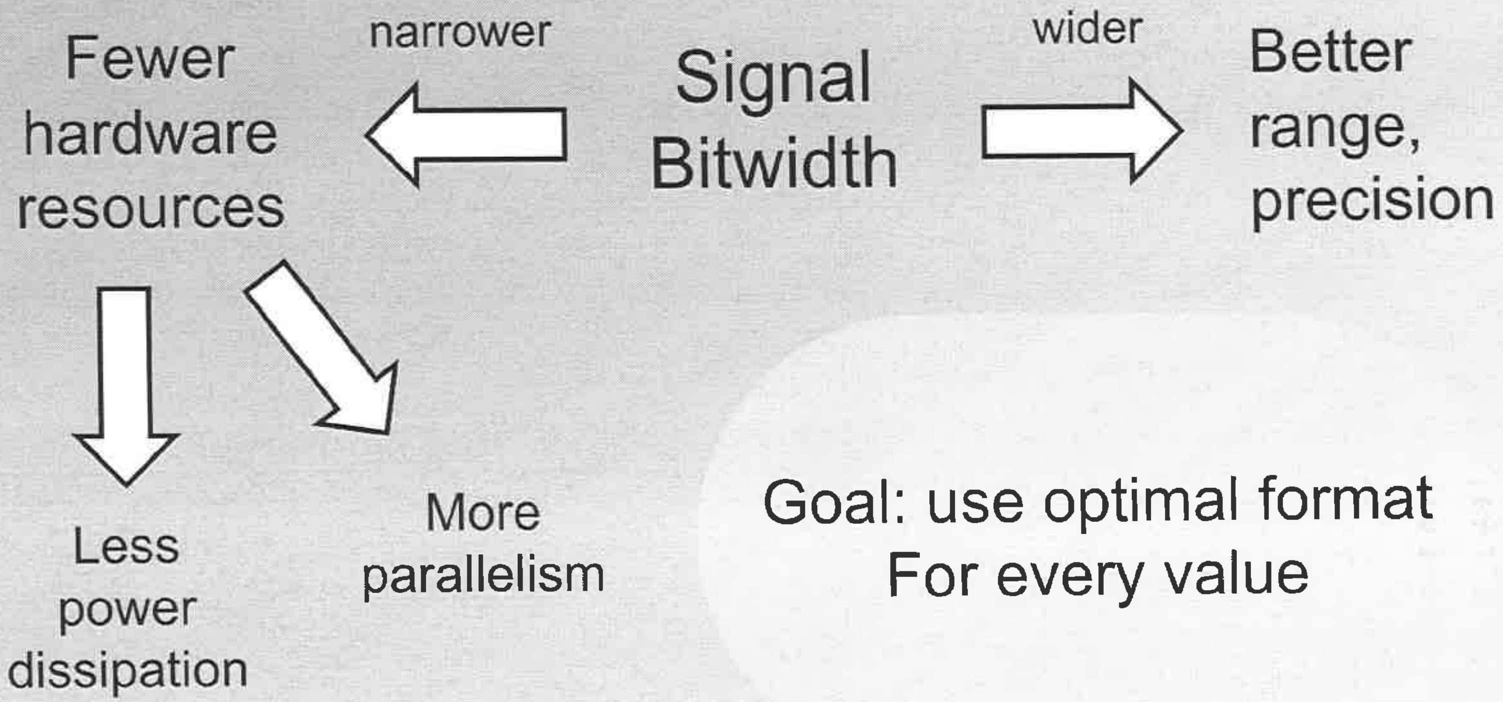
- Introduction and motivation
- Library of hardware modules for variable precision floating point
- Application: K-means algorithm using variable precision floating point library
- Conclusions



Accelerating Algorithms

- Reconfigurable hardware used to accelerate image and signal processing algorithms
- Exploit parallelism for speedup
- Customize design to fit the particular task
  - signals in fixed or floating-point format
  - area, power vs. range, precision trade-offs

Format Design Trade-offs



Fewer hardware resources

narrower

Signal Bitwidth

wider

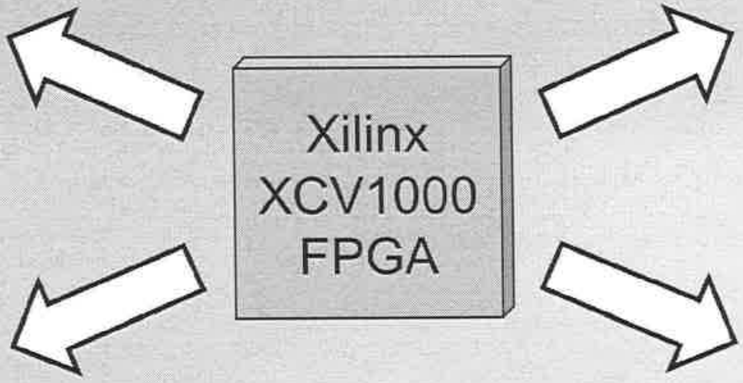
Better range, precision

Less power dissipation

More parallelism

Goal: use optimal format
For every value

Area Gains Using Reduced Precision



IEEE single precision

31 adders

OR

13 multipliers

12-bit floating point format

113 adders

OR

85 multipliers

General Floating-Point Format

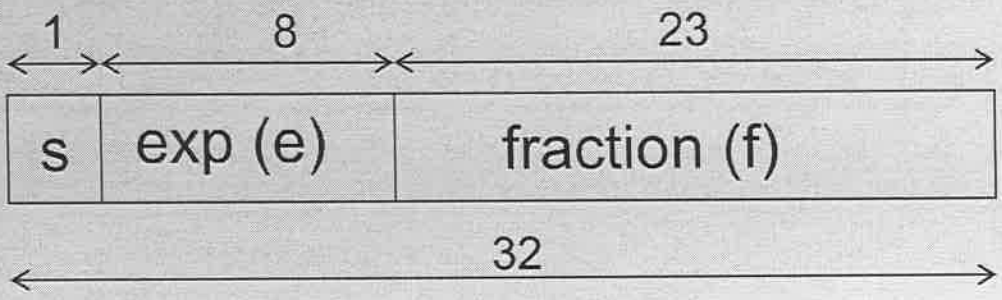
Field	Symbol	Bitwidth
sign	s	1
exponent	e	*exp_bits*
fraction/mantissa	f	*man_bits*

$$(-1)^{s} * 1.f * 2^{e\text{-}BIAS}$$

sign	exponent	fraction / mantissa

MSB LSB

IEEE Floating Point Format



$$(-1)^{s} * 1.f * 2^{e\text{-}BIAS}$$

- BIAS depends on number of exponent bits
  - » 127 in IEEE single precision format
- Implied 1 in mantissa not stored

Library of Parameterized Modules

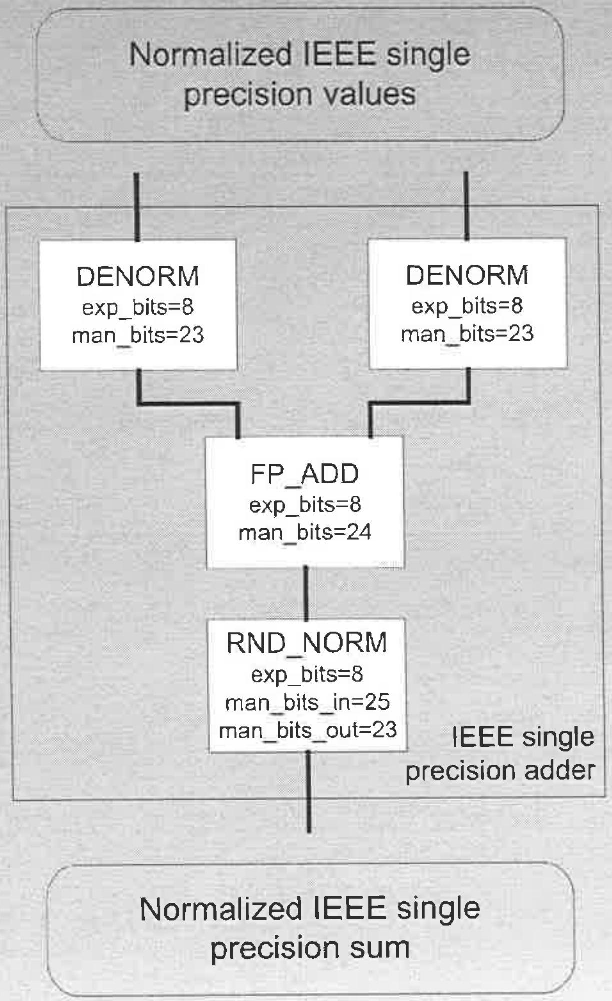
- Total of seven parameterized hardware modules for arbitrary precision floating-point arithmetic

	Module	**Latency**
format control	denorm	0
	rnd_norm	2
operators	fp_add	4
	fp_sub	4
	fp_mul	3
conversion	fix2float	4 / 5
	float2fix	4 / 5

Highlights

- Completely general floating-point format
- All IEEE formats are a subset
- All previously published non-IEEE formats are a subset
- Abstract normalization from other operations
- Rounding to zero or nearest
- Pipelining signals
- Some error handling

Assembly of Modules



Normalized IEEE single precision values

DENORM
exp_bits=8
man_bits=23

DENORM
exp_bits=8
man_bits=23

FP_ADD
exp_bits=8
man_bits=24

RND_NORM
exp_bits=8
man_bits_in=25
man_bits_out=23

IEEE single precision adder

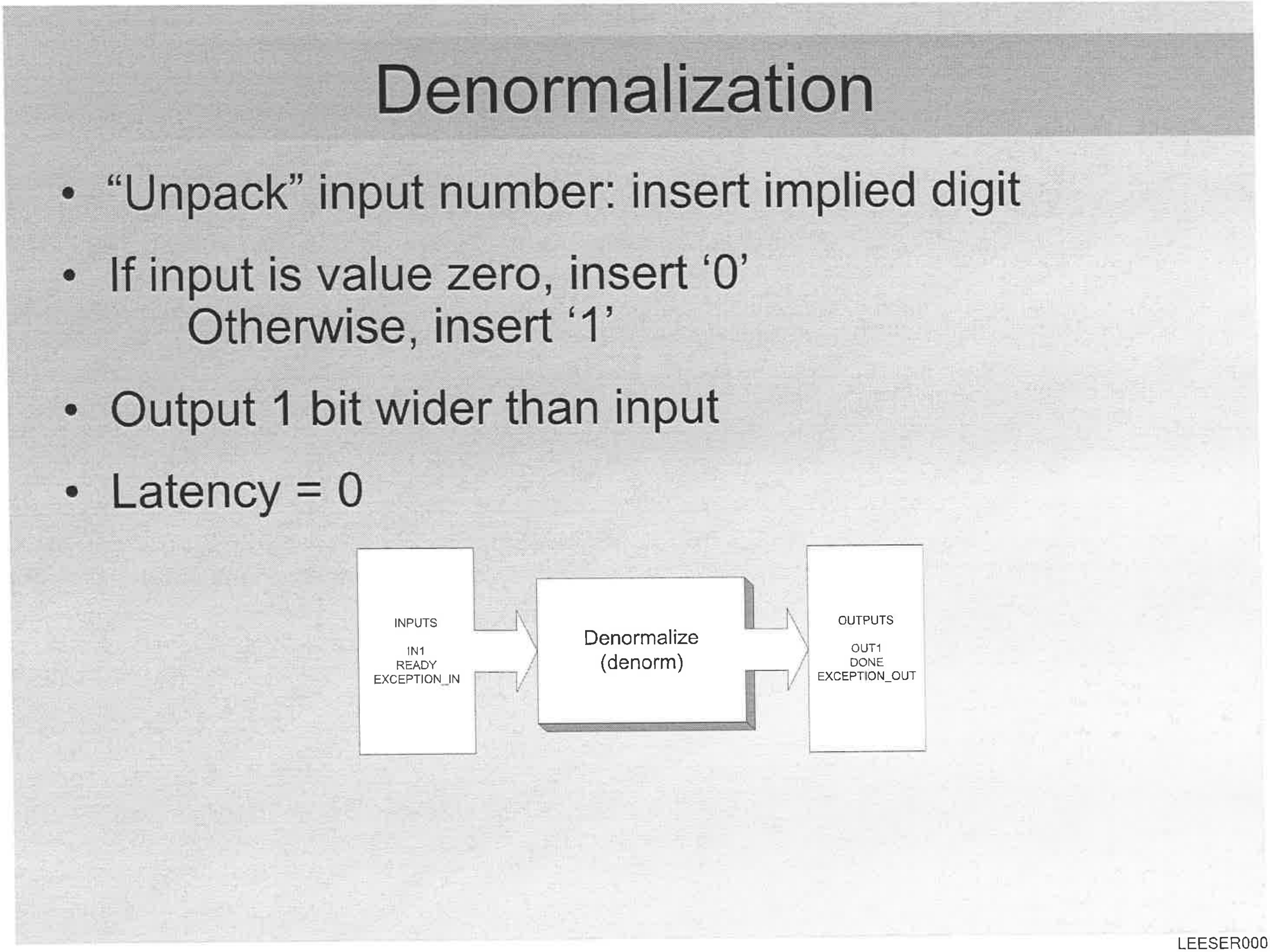
Normalized IEEE single precision sum

2 × denorm

+ 1 × fp_add

+ 1 × rnd_norm

= 1 × IEEE single precision adder



Denormalization

- “Unpack” input number: insert implied digit
- If input is value zero, insert ‘0’
  Otherwise, insert ‘1’
- Output 1 bit wider than input
- Latency = 0

INPUTS

IN1
READY
EXCEPTION_IN

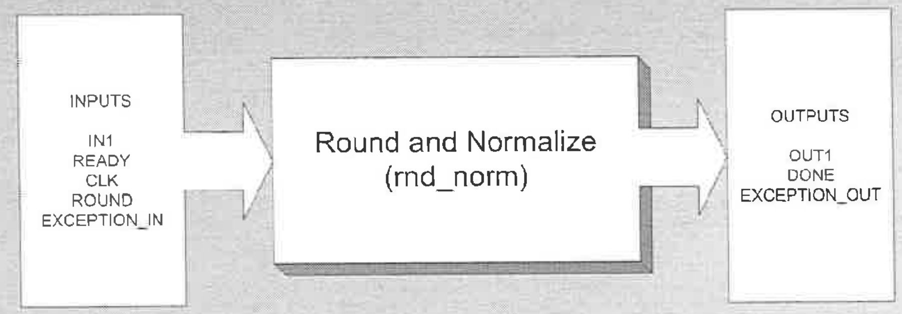
Denormalize
(denorm)

OUTPUTS

OUT1
DONE
EXCEPTION_OUT

Rounding and Normalizing

- Returns input to normalized format
- Designed to follow arithmetic operation(s)



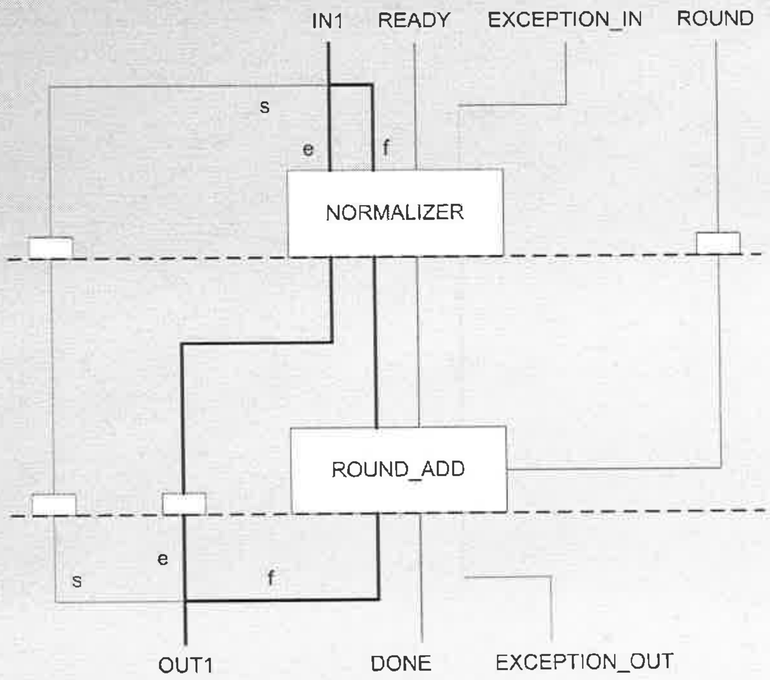
INPUTS

IN1
READY
CLK
ROUND
EXCEPTION_IN

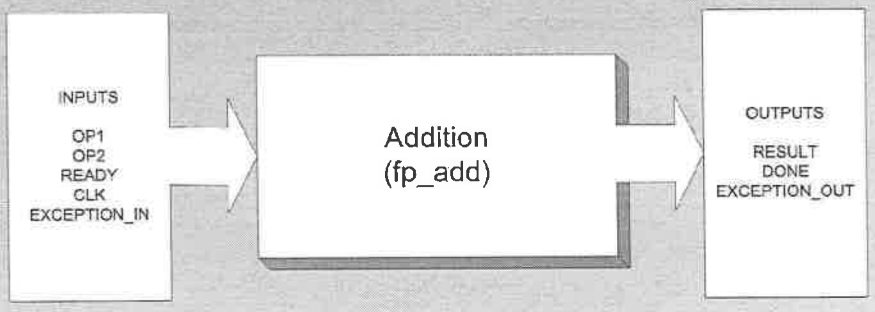
Round and Normalize
(rnd_norm)

OUTPUTS

OUT1
DONE
EXCEPTION_OUT



Addition and Subtraction



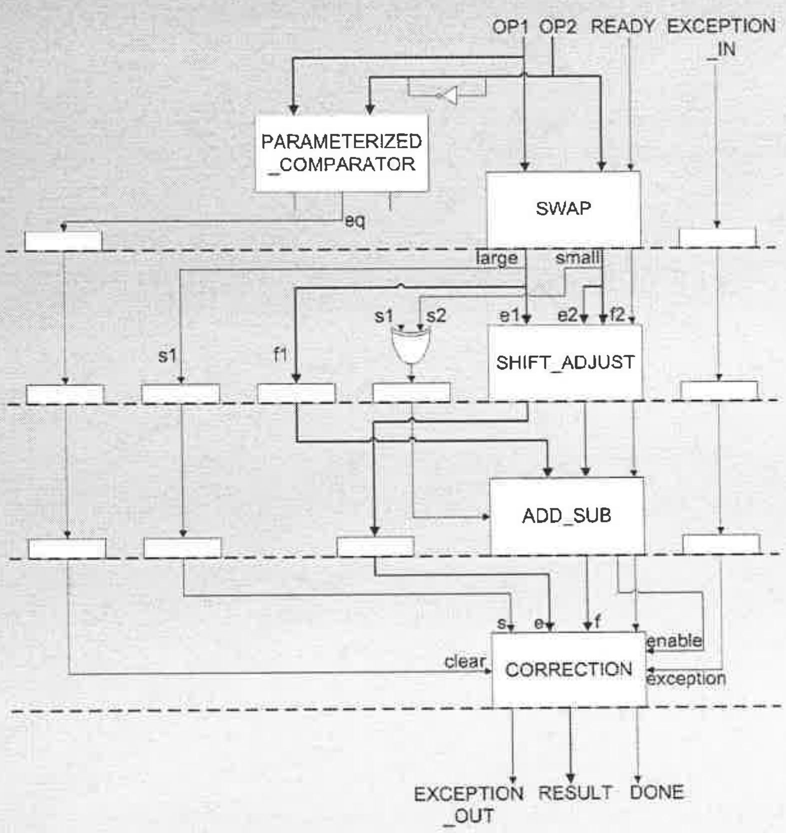
INPUTS

OP1
OP2
READY
CLK
EXCEPTION_IN

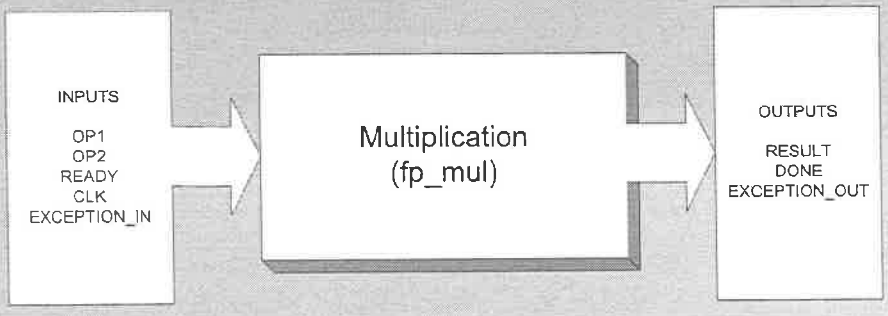
Addition
(fp_add)

OUTPUTS

RESULT
DONE
EXCEPTION_OUT



Multiplication

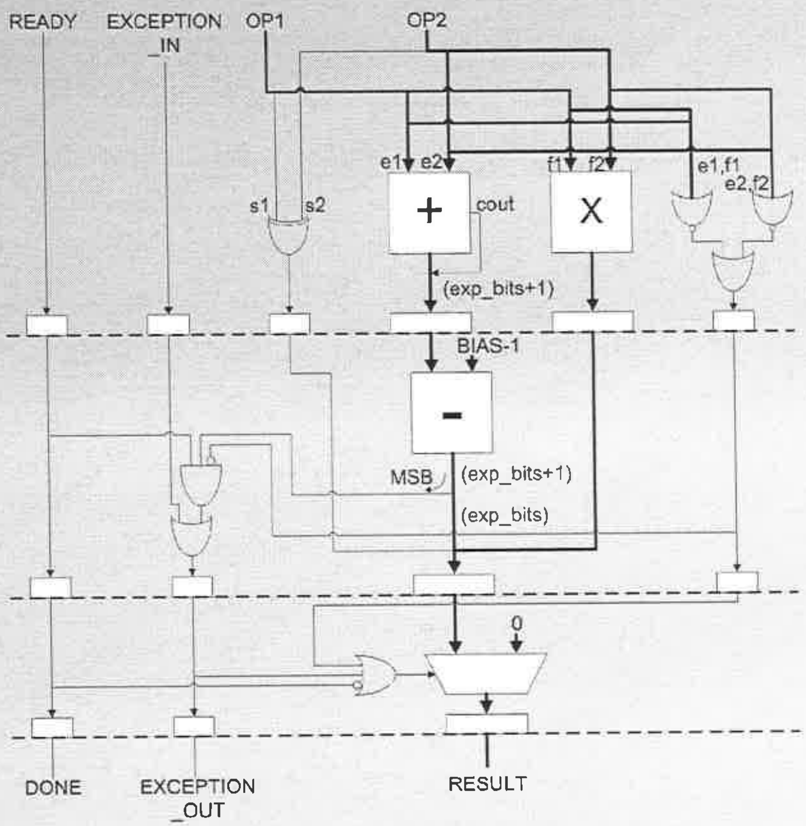

INPUTS

OP1
OP2
READY
CLK
EXCEPTION_IN

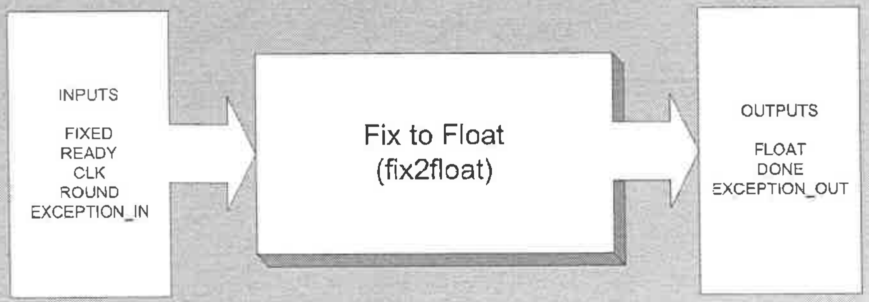
Multiplication
(fp_mul)

OUTPUTS

RESULT
DONE
EXCEPTION_OUT



Fixed to Floating-Point



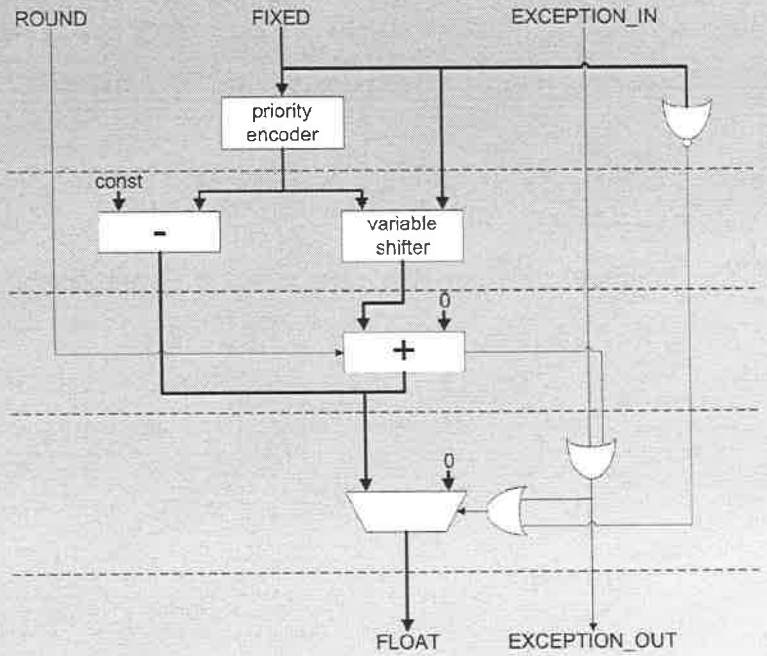
INPUTS

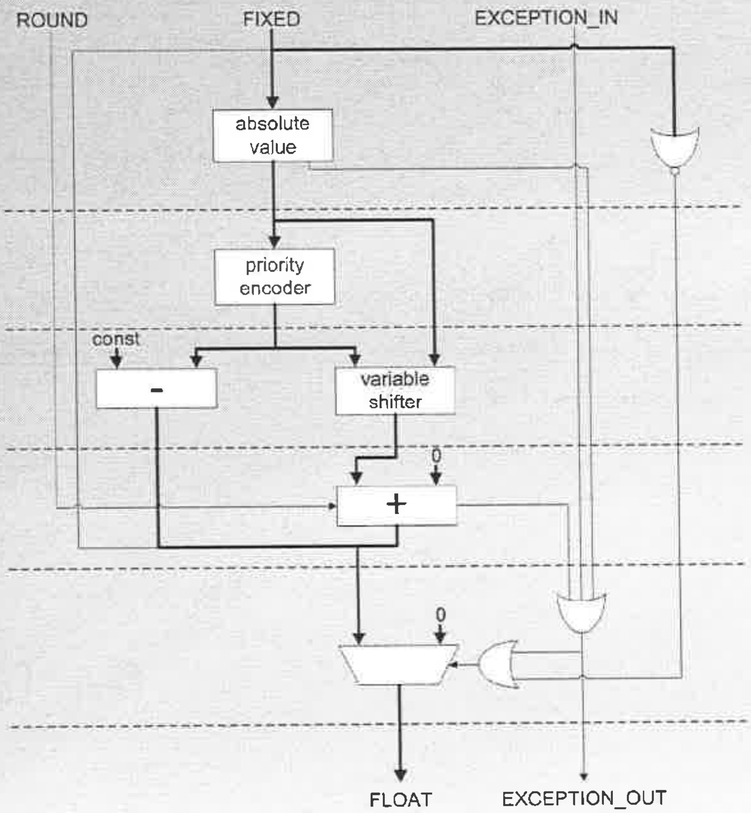
FIXED
READY
CLK
ROUND
EXCEPTION_IN

Fix to Float
(fix2float)

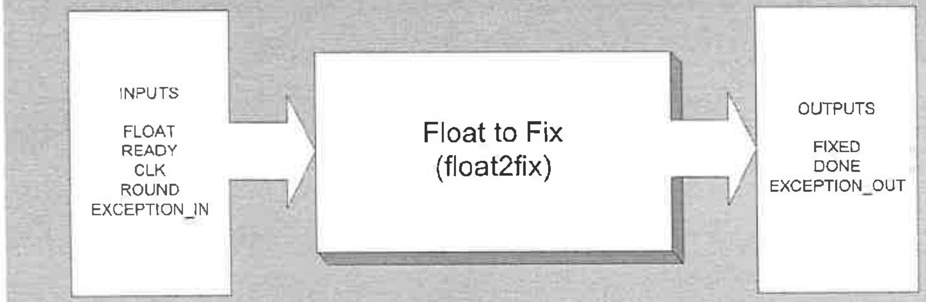
OUTPUTS

FLOAT
DONE
EXCEPTION_OUT





Floating to Fixed-Point



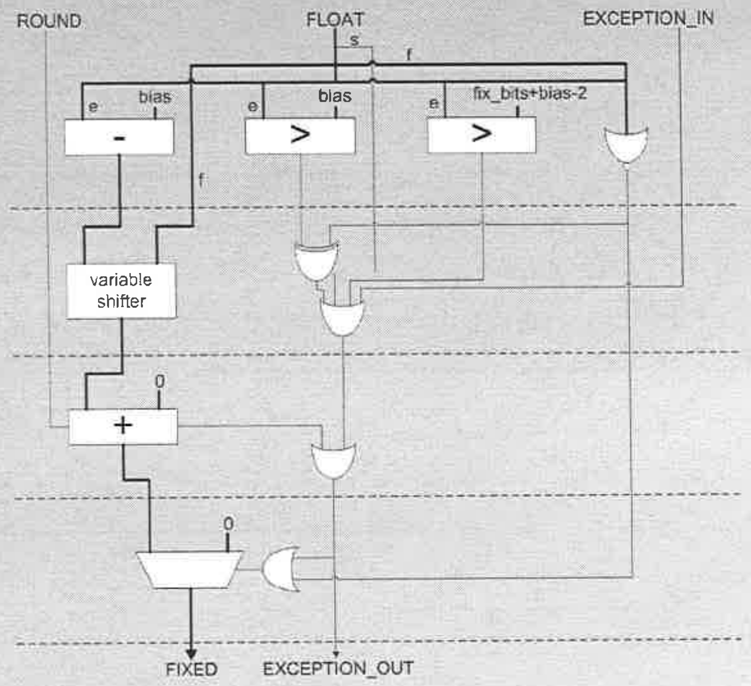
INPUTS

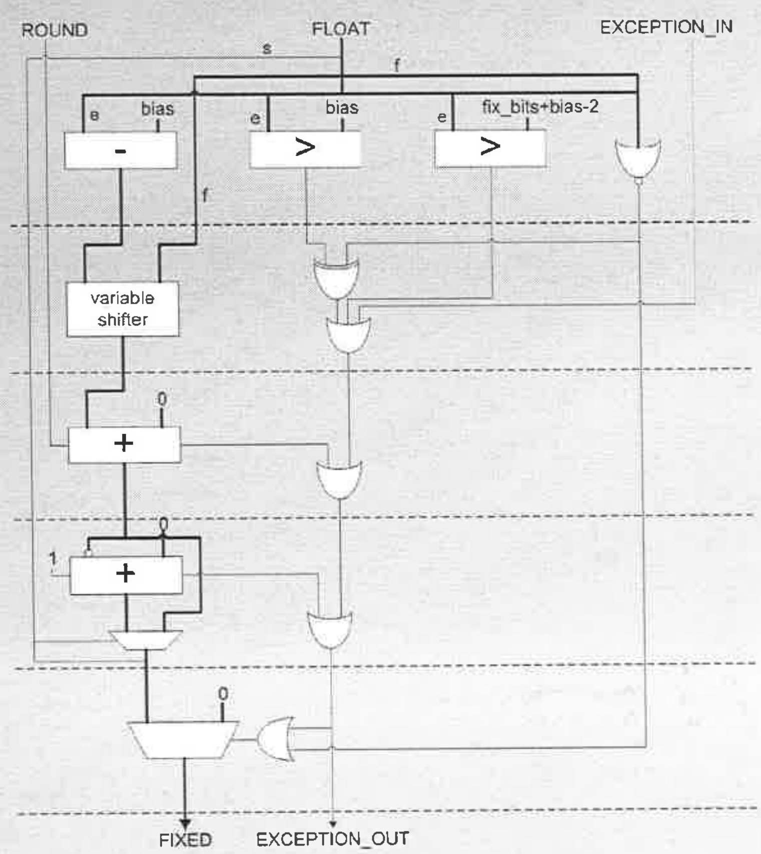
FLOAT
READY
CLK
ROUND
EXCEPTION_IN

Float to Fix
(float2fix)

OUTPUTS

FIXED
DONE
EXCEPTION_OUT

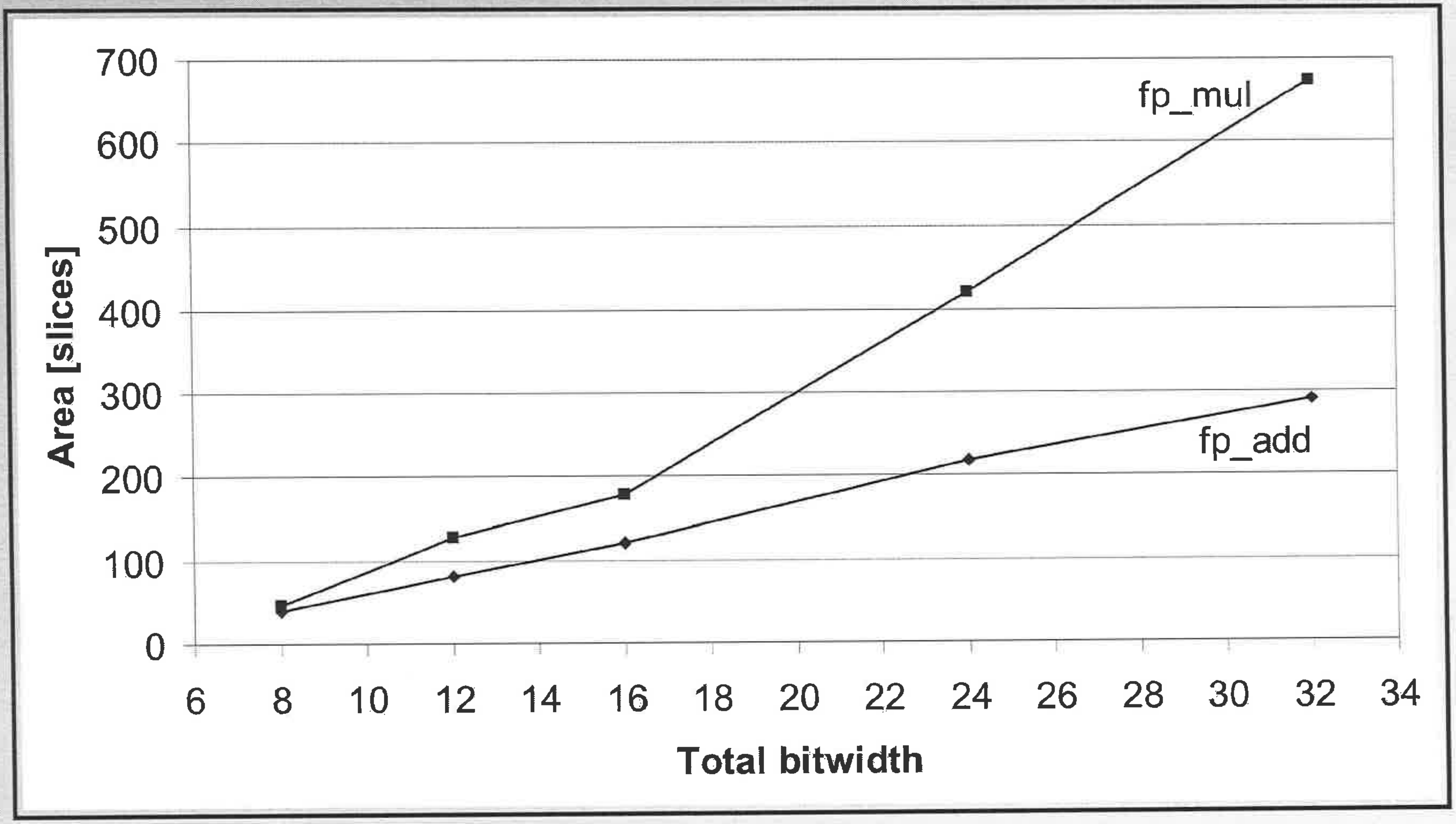




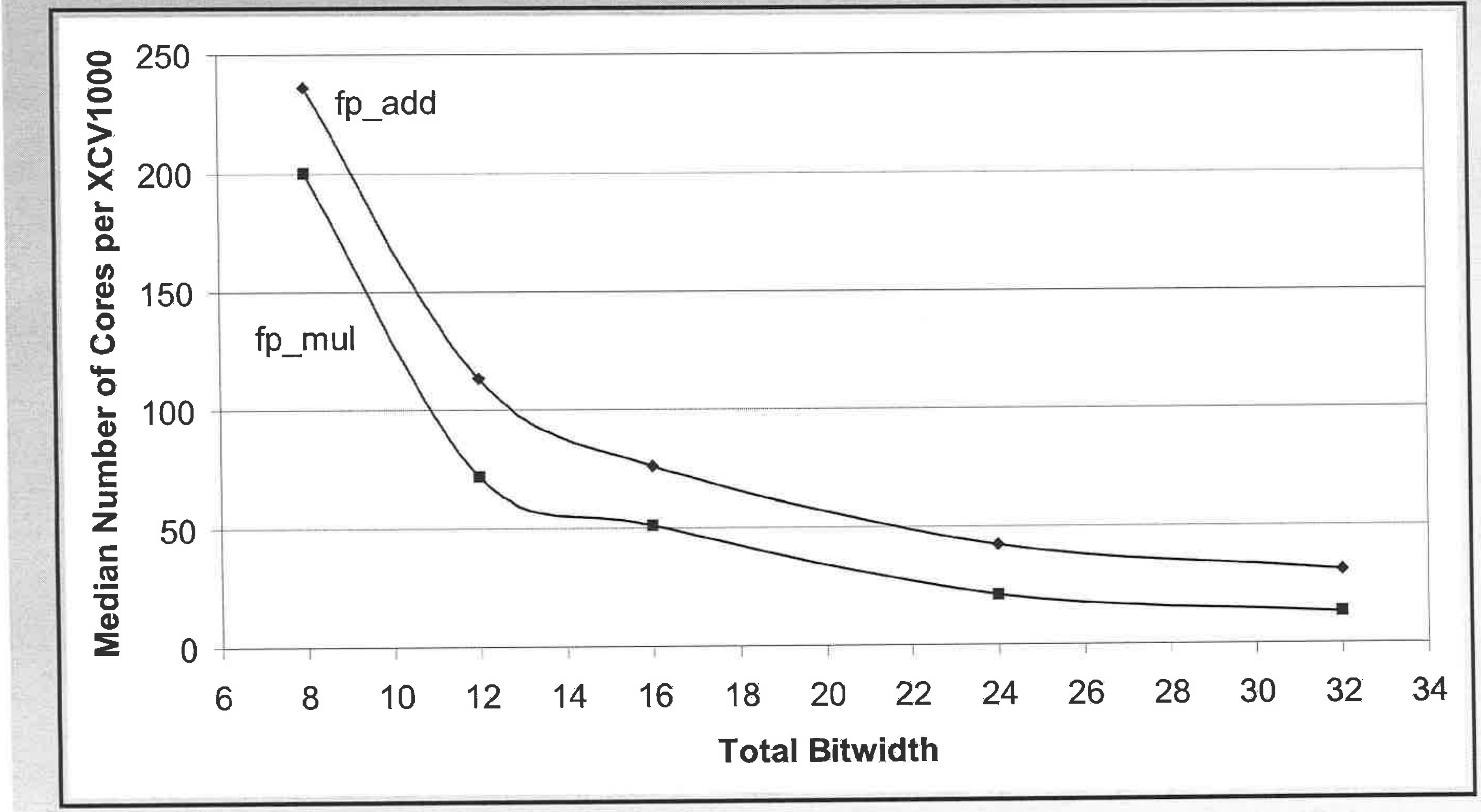
Implementation Experiments

- Designs specified in VHDL
- Mapped to Xilinx Virtex FPGA
- Wildstar reconfigurable computing engine by Annapolis Micro Systems Inc.
  - PCI Interface to host processor
  - 3 Xilinx XCV1000 FPGAs
  - total of 3 million system gates
  - 40 Mbytes of SRAM
  - 1.6 Gbytes/sec I/O bandwidth
  - 6.4 Gbytes/sec memory bandwidth
  - clock rates to 100MHz

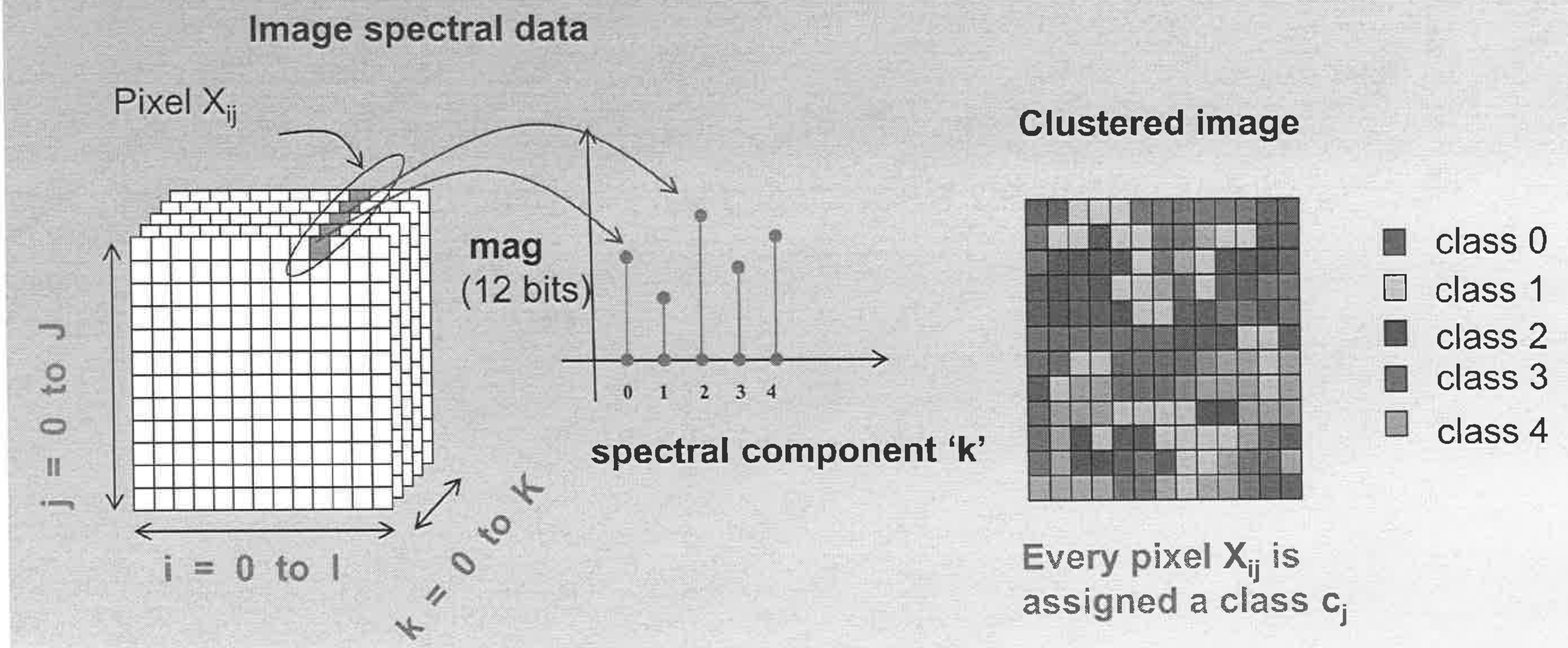
Synthesis Results



Synthesis Results



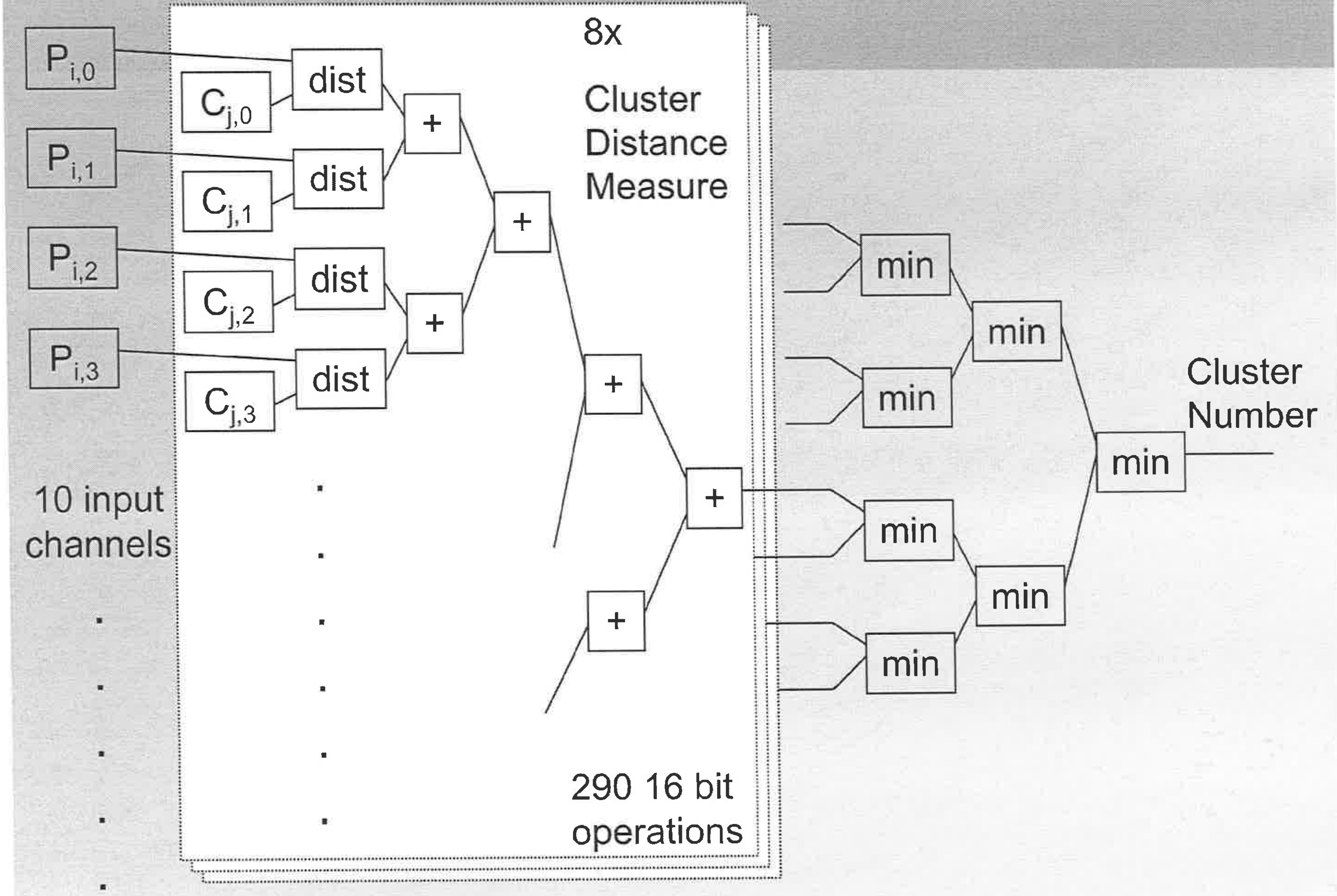
K-means Algorithm



Every pixel X_{ij} is assigned a class c_j

- Each cluster has a center:
  - – mean value of pixels in that cluster
- Each pixel is in the cluster whose center it is closest to
  - – requires a distance metric
- Algorithm is iterative

Hardware Implementation of k-means Clustering



8x

Cluster Distance Measure

10 input channels

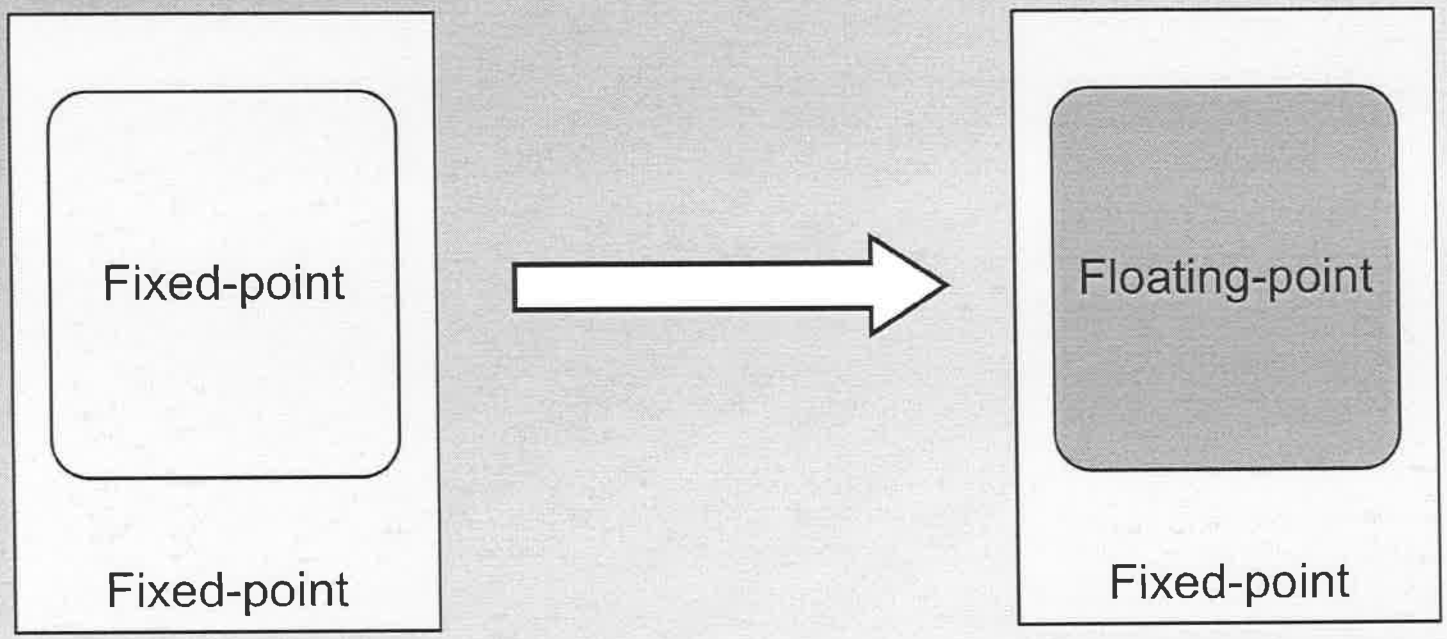
290 16 bit operations

Cluster Number

K-means Clustering Algorithm

Purely fixed-point

Hybrid fixed and floating-point



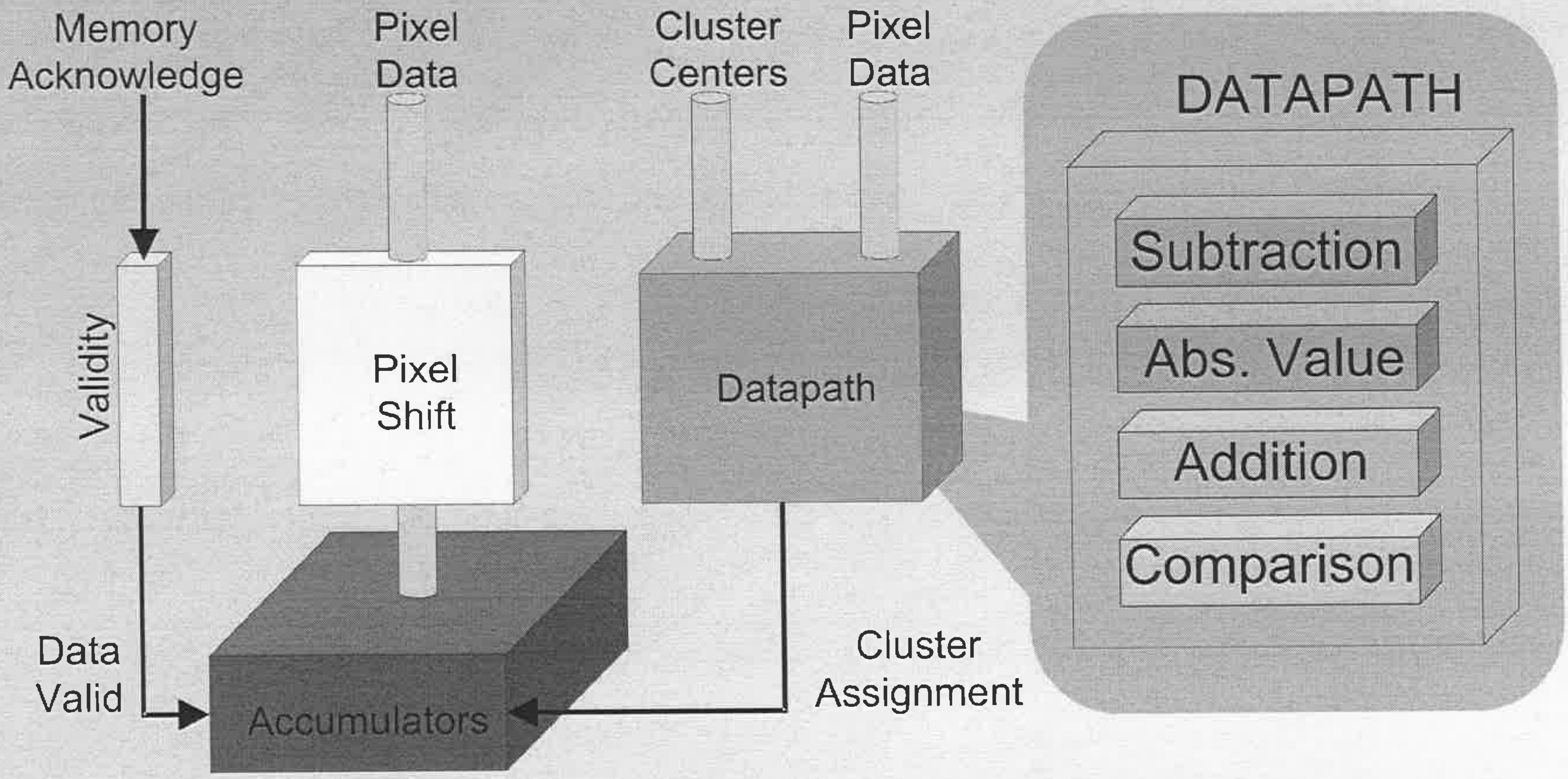
Fixed-point

Fixed-point

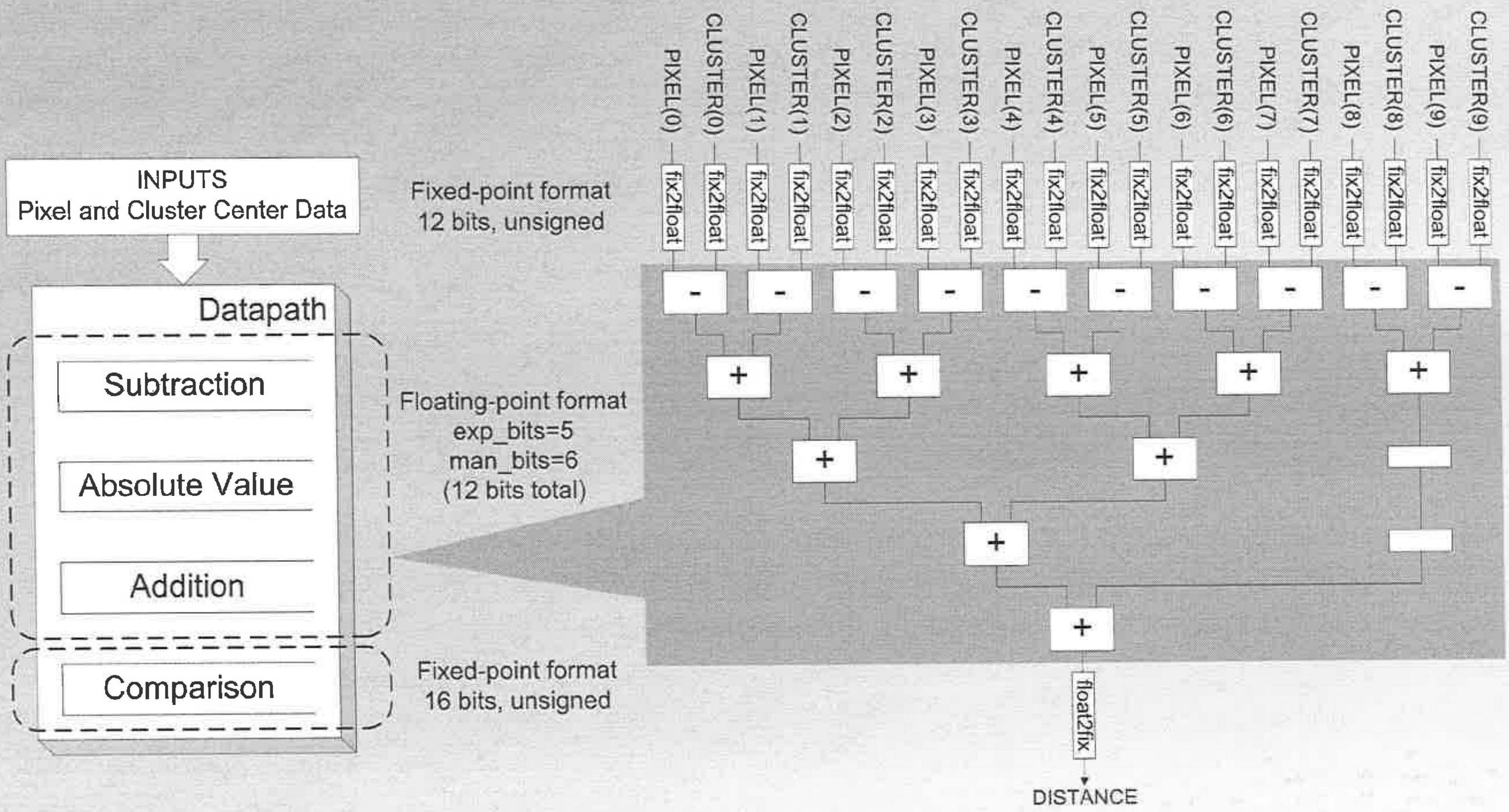
Floating-point

Fixed-point

Structure of the K-means Circuit



Hybrid Datapath



INPUTS
Pixel and Cluster Center Data

Datapath

Subtraction

Absolute Value

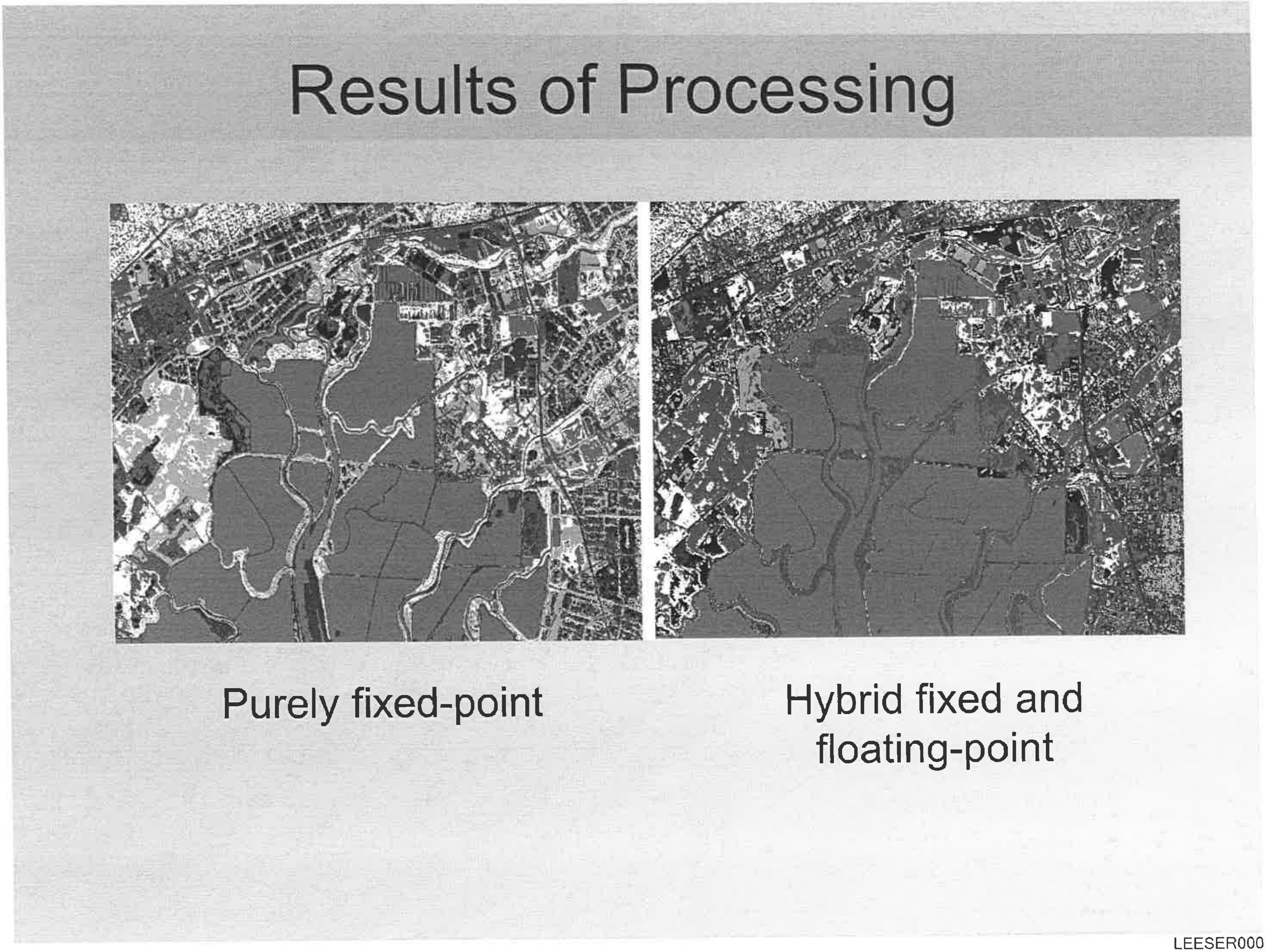
Addition

Comparison

Fixed-point format
12 bits, unsigned

Floating-point format
exp_bits=5
man_bits=6
(12 bits total)

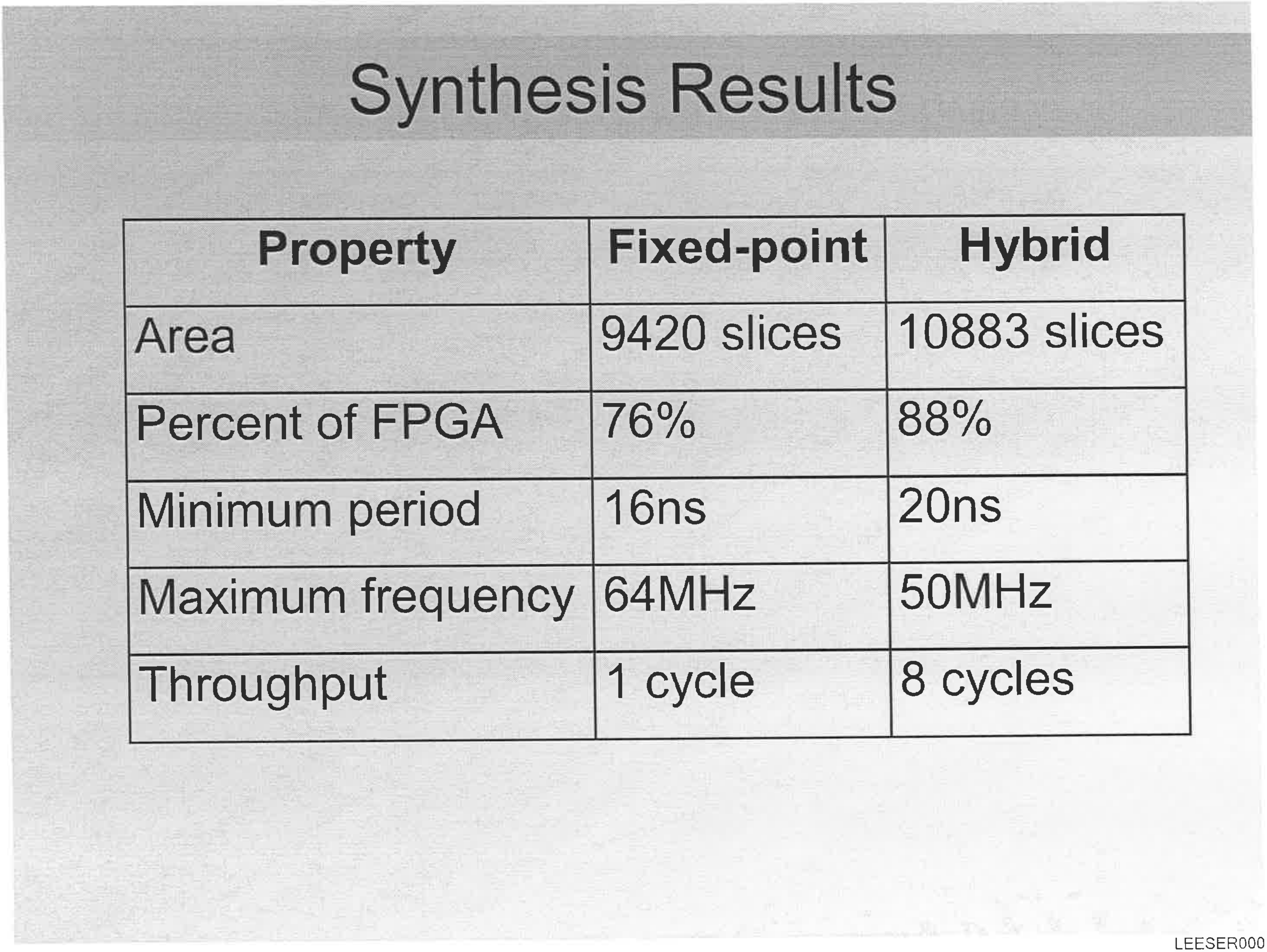
Fixed-point format
16 bits, unsigned



Results of Processing

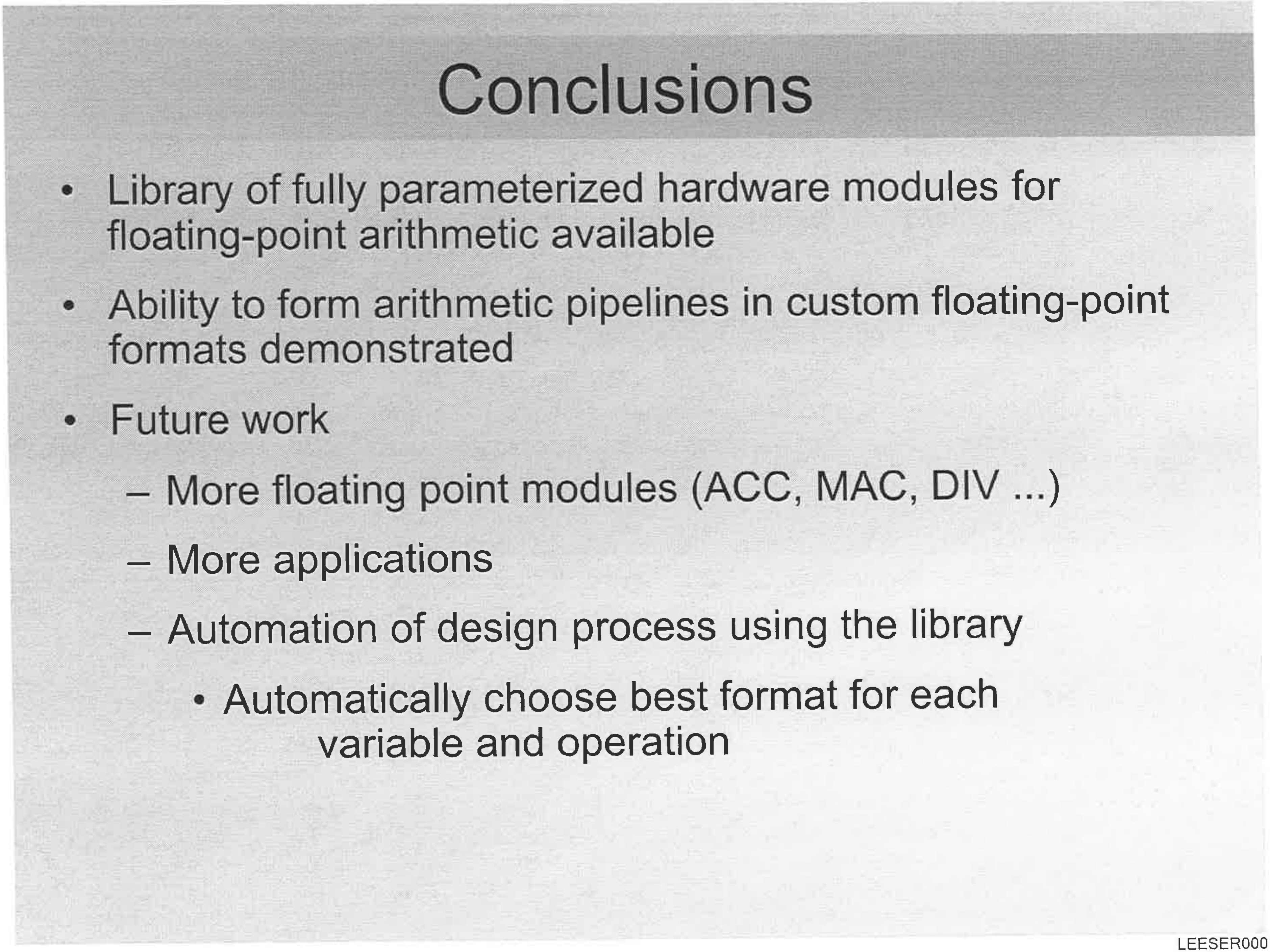
Purely fixed-point

Hybrid fixed and floating-point



Synthesis Results

Property	Fixed-point	Hybrid
Area	9420 slices	10883 slices
Percent of FPGA	76%	88%
Minimum period	16ns	20ns
Maximum frequency	64MHz	50MHz
Throughput	1 cycle	8 cycles



Conclusions

- Library of fully parameterized hardware modules for floating-point arithmetic available
- Ability to form arithmetic pipelines in custom floating-point formats demonstrated
- Future work
  - More floating point modules (ACC, MAC, DIV ...)
  - More applications
  - Automation of design process using the library
    - Automatically choose best format for each variable and operation